Towards a Higher Order Unification Based on Proof Nets(Type Theory and its Applications to Computer Systems)

AUTHOR(S):

Matsuoka, Satoshi

CITATION:

Matsuoka, Satoshi. Towards a Higher Order Unification Based on Proof Nets(Type Theory and its Applications to Computer Systems). 数理解析研究所講究録 1998, 1023: 1-15

ISSUE DATE:

1998-01

URL:

http://hdl.handle.net/2433/61724

数理解析研究所講究録
1023巻 1998年 1-15

Towards a Higher Order Unification Based on Proof Nets

Satoshi Matsuoka

Department of Electrical and Computer Engineering, Faculty of Engineering,
Nagoya Institute of Technology, Gokiso, Showa-Ku, 466, Japan.
matsuoka@juno.ics.nitech.ac.jp

Abstract. We study a higher order unification procedure based on proof nets. In this paper, we present a higher order pre-unification algorithm for intuitionistic implicational multiplicative fragment of proof nets and show termination, soundness and completeness for the algorithm. It is discussed that an extension to exponential connectives can be easily done. Moreover, comparisons of Cerversalo and Pfenning's work are also discussed.

1 Introduction

Higher-order unification has been studied in many literatures, e.g., [Hue75, SG89, Wol93]. As a part of implementations of theorem provers, it has been used [Pau86](more precisely, higher-order *pre-unification* algorithm has been used). Higher-order unification is a procedure that when two lambda terms s and t are given, find a substitution θ for terms such that θs and θt is equal under a equality relation. The idea of the procedure is described as follows : at first, try to unify the heads of s and t, and if it succeeds, then try to unify all arguments of s and t.
In this paper, we study higher order unification procedure based on proof nets. In this paper, we only consider intuitionistic fragment of MELL proof net by using Danos-Regnier polarity. An attempt to restrict higher order unification to a weaker fragment was done in [Mil91]. This system is called L_λ. Our study is also a similar attempt. While higher order unification on L_λ is a decidable and deterministic system, our higher order unification without exponential connectives is a decidable and nondeterministic system.
Recently, higher order unification via explicit substitutions has been proposed [DHK95]. Obviously our system is simpler than that w.r.t the definition of substitutions and unification procedure and like their system our system with exponential connectives can also simulate higher order unification based on simply typed lambda calculus (via Girard's translation from intuitionistic logic to Linear Logic).

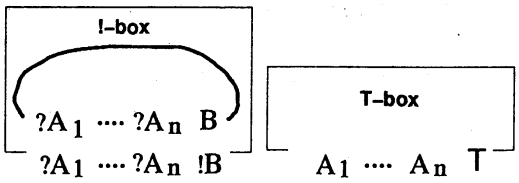
2 IIMELL proof nets

In this section, we introduce proof nets for implicational intuitionistic multiplicative exponential fragment with $\top$ of Linear Logic with (IIMELL). IIMELL

without exponential connectives is called IIMLL.
We assume that readers are familiar with proof nets of the multiplicative exponential fragment of Linear Logic (MELL). As references, we list [Gir95b, Laf95]. We only list links which we use in this paper.

ID-links $\dfrac{}{A \quad A^{\perp}}$ Cut links $\dfrac{A \quad A^{\perp}}{\mathrm{Cut}}$ times links $\dfrac{A \quad B}{A \otimes B}$ par links $\dfrac{A \quad B}{A \wp B}$

d-links $\dfrac{A}{?A}$ w-links $?A$ c-links $\dfrac{?A \cdots ?A}{?A}$



Definition 1 (polarized formulas). A formula of IIMELL is a pair $\langle A, p\rangle$ where A is a formula of MELL in the usual sense and p is a element of $\{+,-\}$. $+$ and $-$ are called *Danos-Regnier polarity*. A formula $\langle A, p\rangle$ can be written as A^p. A formula of IIMELL is called polarized formula. A formula with $+$ (resp. $-$) polarity is called $+$-formula or positive formula (resp. $-$-formula or negative formula).

Definition 2. For each formula X^+ (or X^-) of IIMELL, we define its dual formula $(X^+)^*$ (or $(X^-)^*$) inductively as follows (notice that we omit the definition of the dual formulas that we do not use in IIMELL proof nets actually):

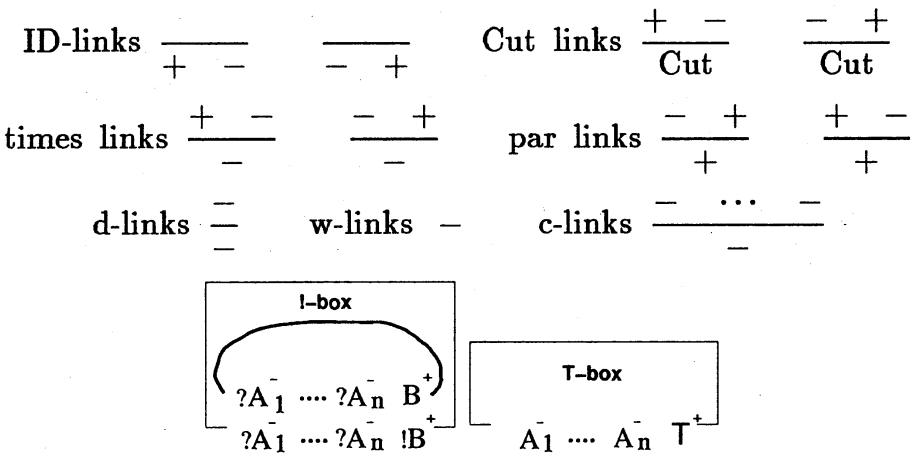
1. If p is an atomic formula (in the usual sense), then $(p^+)^* \equiv (p^{\perp})^-$ and $((p^{\perp})^-)^* \equiv p^+$;
2. $((B \otimes C)^-)^* \equiv (B^* \wp C^*)^+$;
3. $((B \wp C)^+)^* \equiv (B^* \otimes C^*)^-$;
4. $((!B)^+)^* \equiv (?B^*)^-$;
5. $((?B)^-)^* \equiv (!B^*)^+$.

Definition 3. An IIMELL proof net is an MELL proof net $\Theta = (V, E)$ except that

1. V is a multiset of polarized formulas;
2. ID-links and Cut-links are changed to

$$\text{ID-links } \frac{}{A \quad A^*} \qquad \text{Cut links } \frac{A \quad A^*}{\mathrm{Cut}};$$

3. Any signs of formulas in V and links in E must satisfy one of following conditions:



Theorem 4. IIMELL *proof nets are sequentializable w.r.t.* IIMELL *sequent calculus.*

Proof. By using sequentialization theorem for MELL proof nets. □

Proposition 5. *Any* IIMELL *proof net* Θ *has exactly one* $+$*-formula in the multiset of conclusions of* Θ*.*

3 Higher Order Unification Based on IIMELL Proof Nets

3.1 Preliminaries

In this subsection, we define higher order unification problems for IIMELL proof nets.
By Proposition 5 in Section 3, any IIMELL proof net Θ has exactly one $+$-formula in the multiset of the conclusions of Θ.

Definition 1. For any two IIMELL proof nets Θ_1 and Θ_2, if the $+$-formula A^+ in the conclusions of Θ_1 is the same formula as that of Θ_2, then we say that Θ_1 and Θ_2 have the same type A.

Definition 2. $-$-formulas in the conclusions of an IIMELL proof net Θ are called *free ports* of Θ. The $+$-formula in the conclusions of an IIMELL proof net Θ is called *the principal port* of Θ. If Θ does not have any free port, then we say that Θ is closed.

Free ports in Θ may be regarded as free variables in lambda terms.

Definition 3. For each IIMELL formula A^-, we assume given a countably infinite set of names of the formula, denoted $\mathcal{PN}_A$, and let $\mathcal{PN} = \bigcup_{A \text{ is an IIMELL formula}} \mathcal{PN}_A$.
An *object proof net* is an IIMELL proof net Θ with a set of names from $\mathcal{PN}$ such that to each free port in Θ, distinct name from each other is assigned. Let us call the set of names of free ports in an object proof net Θ be $\mathcal{FPN}(\Theta)$.

Definition 4. A substitution θ is a set of IIMELL object proof nets with set of names from $\mathcal{PN}$ such that to each principal port of proof nets in θ, distinct name from each other is assigned.

Definition 5. If Θ is an IIMELL proof net and θ a substitution, then $\theta\Theta$ is defined as follows: if the dual of a free port A^- in Θ is the same type as the principal port B^+ of a proof net Λ in θ and the name of A^- is the same name as that of B^+, then connect Λ with Θ via A^- and B^+ by using a Cut-link.

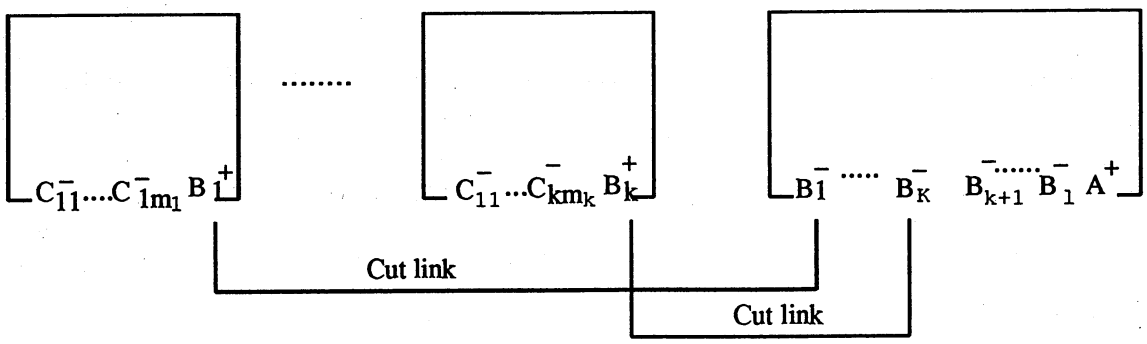


Fig. 1. a substitution on proof nets

Definition 6. A proof net pair is a pair of object proof nets Θ_1 and Θ_2 with the same type denoted by $\langle\Theta_1,\Theta_2\rangle$. A substitution θ is called a unifier of a pair $\langle\Theta_1,\Theta_2\rangle$ if $\theta\Theta_1 = \theta\Theta_2$. A proof net system $S = \{\langle\Theta_1^i,\Theta_2^i\rangle | 1 \leq i \leq n\}$ is a multiset of proof net pairs. A unifier of a proof net system S if it unifies each pair of S.

An unification problem is the problem that find unifiers of a proof net system S.
We assume that for any proof net system $S = \{\langle\Theta_1^i,\Theta_2^i\rangle | 1 \leq i \leq n\}$ from which we start unification procedure, for any $1 \leq i,j \leq n$ and $1 \leq k,\ell \leq 2$, $\mathcal{FPN}(\Theta_k^i) \cap \mathcal{FPN}(\Theta_\ell^j) = \emptyset$.

Notice that a name from $\mathcal{PN}$ must not occur in a proof net more than two times but may occur in a proof net system more than two times.

The equality = w.r.t. IIMELL proof nets we consider is equality modulo ID-reduction, multiplicative-reduction, four exponential reductions — d-reduction, w-reduction, c-reduction and !-reduction — and $\bar{\eta}$-reduction. Moreover, every $\top$-box is regarded as all equal.
Figure 2 represents $\bar{\eta}$-reduction and Figure 3 represents ID-reduction and Multiplicative-reduction. Figure 4 represents exponential-reductions.

When Θ is an IIMELL proof net, let Θ' be an IIMELL proof net obtained by applying the above eight reductions from Θ until the reductions do not apply to Θ' and call Θ' the $\bar{\eta}$-normal form of Θ. The strong normalization theorem for

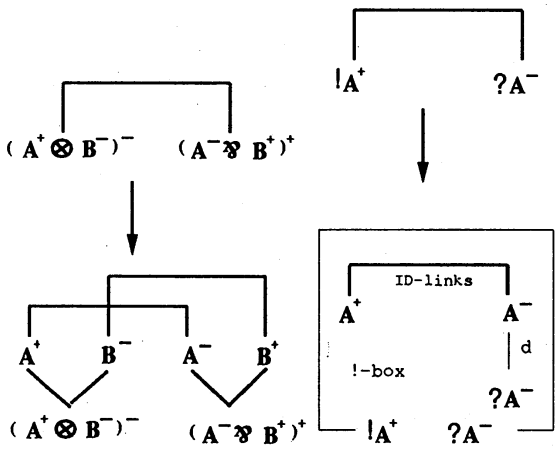


Fig. 2. $\bar{\eta}$-reduction on proof nets

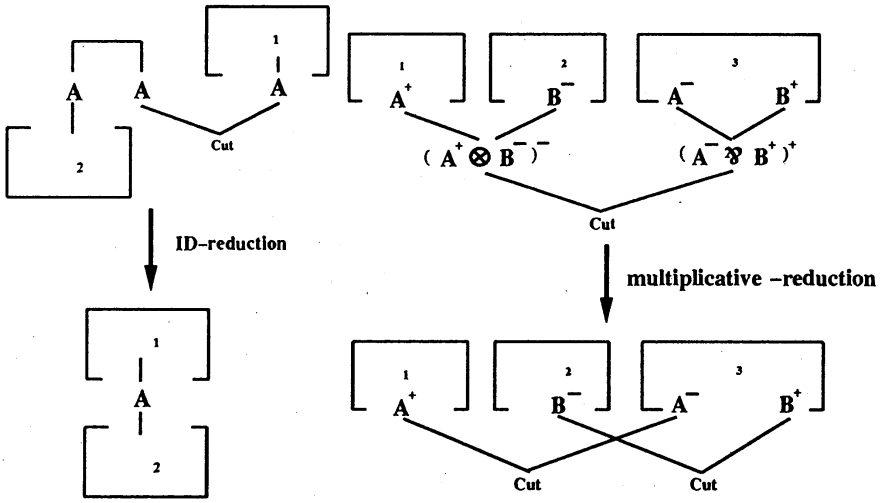


Fig. 3. ID and Multiplicative reductions

this system is obtained from basic proof theory.
As in the same manner of higher order unification based on lambda terms, constants may be considered. Constants are regarded as one-node proof nets:

$$f : A^+$$

The one node proof net must has polarity $+$, and must be labeled by labels that are distinct from $\mathcal{PN}$. We denote labels for constants by f, g and h. In an IIMELL proof net, several constants with the same label may be occurred.
Figure 5 shows any normal form Θ of IIMELL proof nets.

A formula $(X^- \wp Y^+)^+$ is abbreviated by $(X \multimap Y)^+$ and $(X^+ \otimes Y^-)^-$ is abbreviated by $(X \multimap Y)^-$.
In Figure 5,

1. p is an atomic formula.
2. Each e_i $(1 \leq i \leq m)$ is an IIMELL proof net with A_i^+ as the principal port.

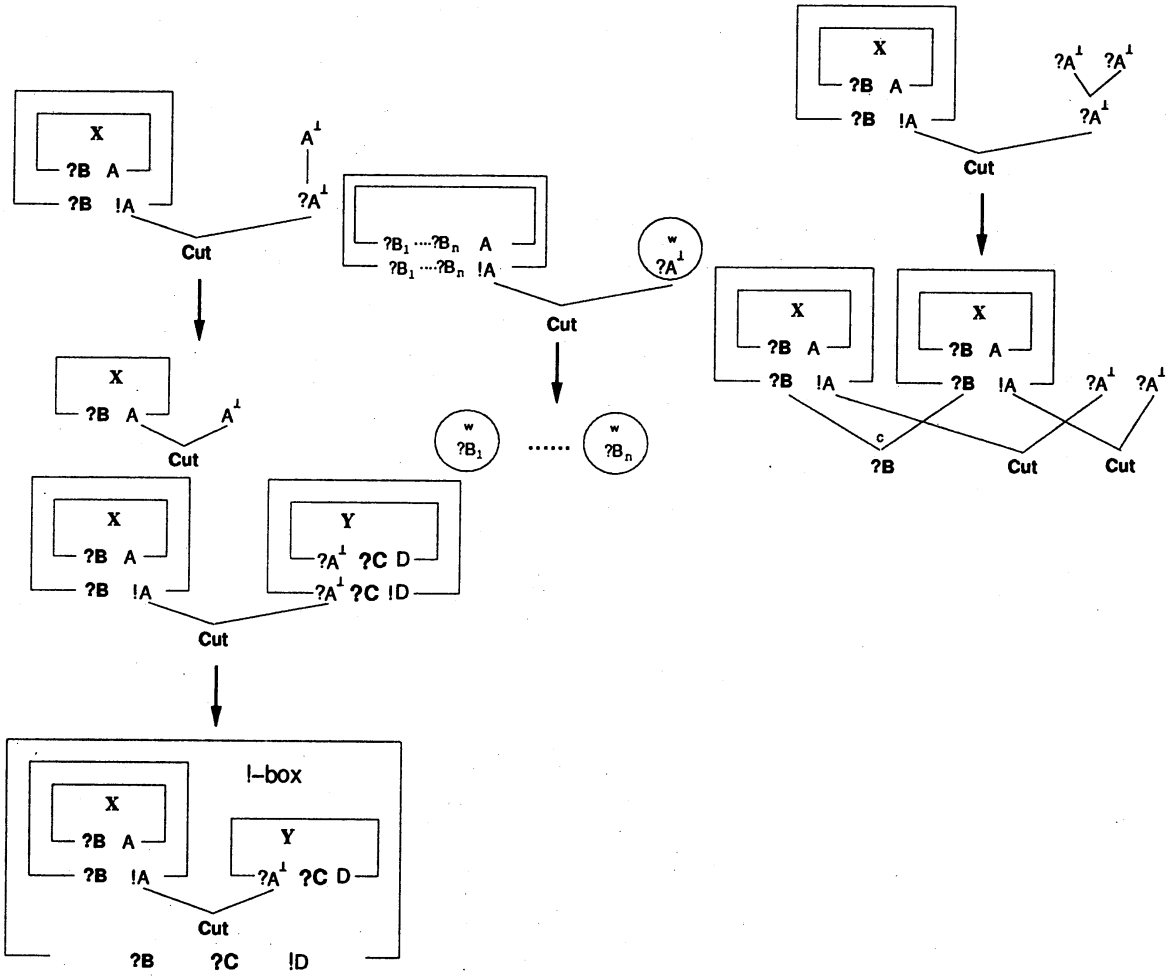


Fig. 4. Exponential reductions

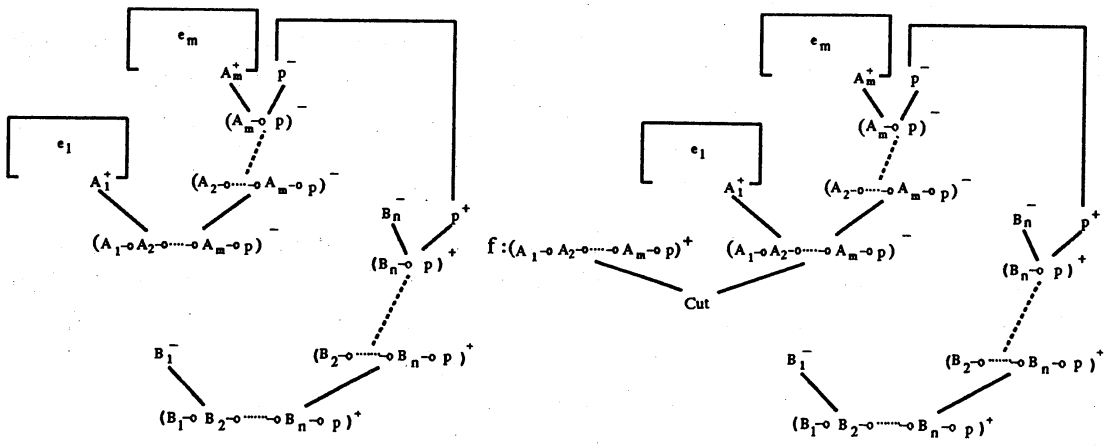


Fig. 5. normal forms of IIMLL proof nets

3. $B_1^-, \ldots, B_n^-$ may occur in $(A_1 \multimap \ldots \multimap A_m \multimap p)^-$ (in only former figure) or in free ports of $e_1, e_2, \ldots$ or e_m. The principal port of Θ is $(B_1 \multimap B_2 \multimap \cdots \multimap B_n \multimap p)^+$.
4. The free ports of Θ are $(A_1 \multimap \ldots \multimap A_m \multimap p)^-$ (in only former figure) and free ports of $e_1, e_2, \ldots, e_m$ except for $B_1^-, \ldots, B_n^-$.
5. $(A_1 \multimap \ldots \multimap A_m \multimap p)^-$ in former figure and $f : (A_1 \multimap \ldots \multimap A_m \multimap p)^+$ in latter figure are called the *head* of Θ. $e_1, \ldots, e_m$ are called *arguments* of Θ.
6. $B_1^-, \ldots, B_n^-$ are called bound ports of Θ. If a bound port B_i $(1 \leq i \leq n)$ is a free port of e_j for some $1 \leq j \leq m$, then we say that B_i belongs to e_j and if B_i is the head $(A_1 \multimap \cdots \multimap A_m \multimap p)^-$ of Θ, we say that B_i belongs to the head of Θ.

Definition 7 (lengths of formulas). The length of an IIMLL formula $\xi = B_1 \multimap \cdots \multimap B_n \multimap p$ is n, where p is an atomic formula. In ξ, B_i $(1 \leq i \leq n)$ is called an argument of ξ.

Definition 8 (the base atomic formulas). The base atomic formula of a formula $A_1 \multimap \cdots \multimap A_n \multimap p$ (where p is an atomic formula) is p.

Definition 9 (rigid and flexible proof nets). If the head of Θ is a bound port or constant, then we say that Θ is a rigid proof net and if it is a free port, then we say that Θ is a flexible proof net.

Definition 10. The depth of a normal proof net Θ is inductively defined as follows:

1. ID-links have depth 1.
2. If Θ has the arguments $e_1, \ldots, e_m$ and the depth of each e_i is d_i, then the depth of Θ is $m + \sum_{1 \leq i \leq m} d_i$.

3.2 Higher Order Unification Algorithm

In this subsection, we describe our higher order unification algorithm.

Definition 11 (partial bindings). We define partial bindings for two proof nets Θ_1 and Θ_2 with the same formulas w.r.t. the principal port (let the formula be $C_1 \multimap \cdots \multimap C_\ell \multimap p$). Let the head of Θ_1 be $A_1 \multimap \cdots \multimap A_{m_1} \multimap p$ and that of Θ_2 be $B_1 \multimap \cdots \multimap B_{m_2} \multimap p$.

Partial bindings are classified in the following way:

1. flexible-flexible imitation partial bindings:
   If both heads of Θ_1 and Θ_2 are free ports, then the partial bindings can be available.
   For each of Θ_1 and Θ_2 the unique partial binding is determined: $\top$-box proof net.
   Let the unique partial binding of Θ_1 (resp. Θ_2) be θ_1 (resp. θ_2).
   The principal port of θ_1 (resp. θ_2) is $A_1 \multimap \cdots \multimap A_{m_1} \multimap p$ (resp. $B_1 \multimap \cdots \multimap B_{m_2} \multimap p$).
   Figure 6 show θ_1 and θ_2.

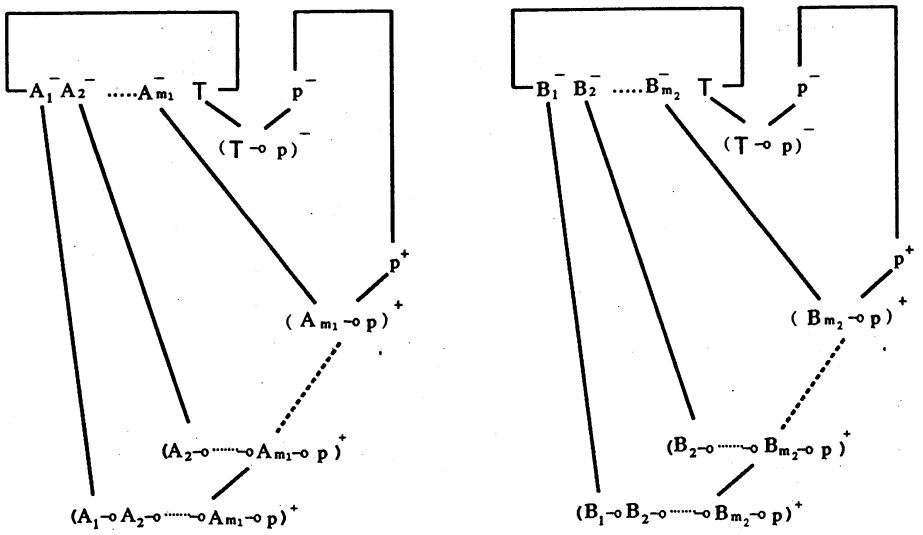


Fig. 6. partial bindings constructed by using $\top$-boxes for Θ_1 and Θ_2

2. flexible-rigid imitation partial bindings:
   If one head of Θ_1 or Θ_2 is a free port and the other head is a constant, then the partial bindings can be available. Let Θ_2 have a constant head (let the constant be f). Then partial bindings θ for Θ_1 are constructed non-deterministically in the following way. The heads of θ must be the same formula as the head of Θ_2 and The principal port of θ must be the dual of the head of Θ_1. Moreover for the arguments $e_1, \cdots, e_n$ of the partial binding (then, the length of the head is n), free port of e_i is only the head of e_i and the free ports have new names from $\mathcal{PN}$ and for the other ports of e_i, any bound ports of the partial binding must belong to them and the arguments of e_i must be $\bar{\eta}$-normal forms of ID-links.
   Figure 7 shows any flexible-rigid imitation partial binding.
3. projection partial bindings:
   If a head $A_1 \multimap \cdots \multimap A_m \multimap p$ of Θ_1 or Θ_2 is a free port (let the proof net be Θ_1) and in the head, a argument A_i is a form $C_1 \multimap \cdots \multimap C_k \multimap p$, then the partial binding can be available. In this case, partial bindings θ are substituted for the head of Θ_1. θ is a proof net with $(A_1 \multimap \cdots \multimap A_m \multimap p)^+$ as the principal port such that the head of θ must be the form $C_1 \multimap \cdots \multimap C_k \multimap p$ and the i-th bound port of θ must belong to the head and for the arguments $e_1, \cdots, e_n$ of the partial binding (then, the length of the head is n), free port of e_i is only the head of e_i and the free ports have new names from $\mathcal{PN}$ and for the other ports of e_i, any bound ports of the partial binding must belong to them and the arguments of e_i must be the $\bar{\eta}$-normal forms of ID-links.
   Figure 8 shows any projection partial binding.

Definition 12 (SIMPLE procedure). We define SIMPLE procedure for the pair of two proof nets $\langle \Theta_1, \Theta_2 \rangle$ with the same head (let the head be $A_1 \multimap \cdots \multimap A_m \multimap p$) and the same principal port. The result of SIMPLE procedure for $\langle \Theta_1, \Theta_2 \rangle$ is denoted by SIMPLE($\langle \Theta_1, \Theta_2 \rangle$). Let the principal port of Θ_1 (resp. Θ_2) be

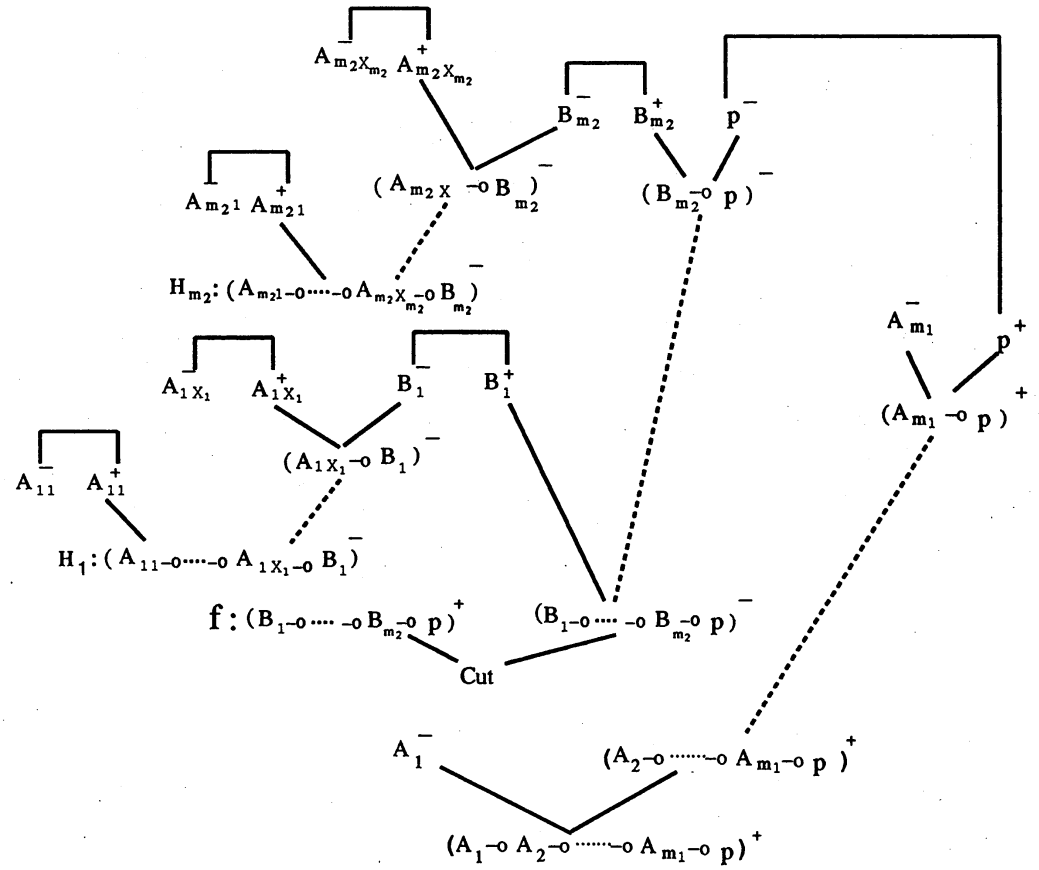


Fig. 7. a flexible-rigid imitation partial binding θ for Θ_1

$B_1 \multimap \cdots \multimap B_n \multimap p$ (resp. $C_1 \multimap \cdots \multimap C_n \multimap p$) (of course, though $B_i = C_i$ as IIMLL formulas, since B_i and C_i may play different roles in Θ_1 and Θ_2, we named them different names). B_i (resp. C_i) is a bound port of Θ_1 (resp. Θ_2). Let the arguments of Θ_1 (resp. Θ_2) be $e_1, \ldots, e_m$ (resp. $f_1, \ldots, f_m$) and let the principal port of e_i and f_i be A_i^+. If Θ_1 and Θ_2 satisfy the following condition, then SIMPLE procedure applies: if any bound port B_i belongs to e_j for some j, then C_i must belongs to f_j, vice versa and if any bound port B_i belongs to the head of Θ_1, then C_i must belongs to the head of Θ_2, vice versa. If Θ_1 and Θ_2 satisfy the above condition, then the following SIMPLE procedure applies:

$$\langle \Theta_1, \Theta_2 \rangle \Longrightarrow \bigcup_{1 \leq i \leq m} \{\langle e_i', f_i' \rangle\}$$

where e_i' (resp. f_i') is the proof net which is constructed by connecting any bound port in Θ_1 (resp. Θ_2) which belongs to e_i (resp. f_i) by using $\wp$-link and by an appropriate order in e_i (resp. in f_i). Otherwise, SIMPLE($\langle \Theta_1, \Theta_2 \rangle$) returns failure.

Definition 13. Let A be an IIMELL formula and F be an free port name from $\mathcal{PN}$. Let Λ be the ID-link that has A^+ and A^- as conclusions and A^- has F as the free port name. Let us call $\bar{\eta}$-normal form of Λ $\bar{\eta}(A, F)$. In the below algorithm, we identify a name of a free port with the formula with the name.

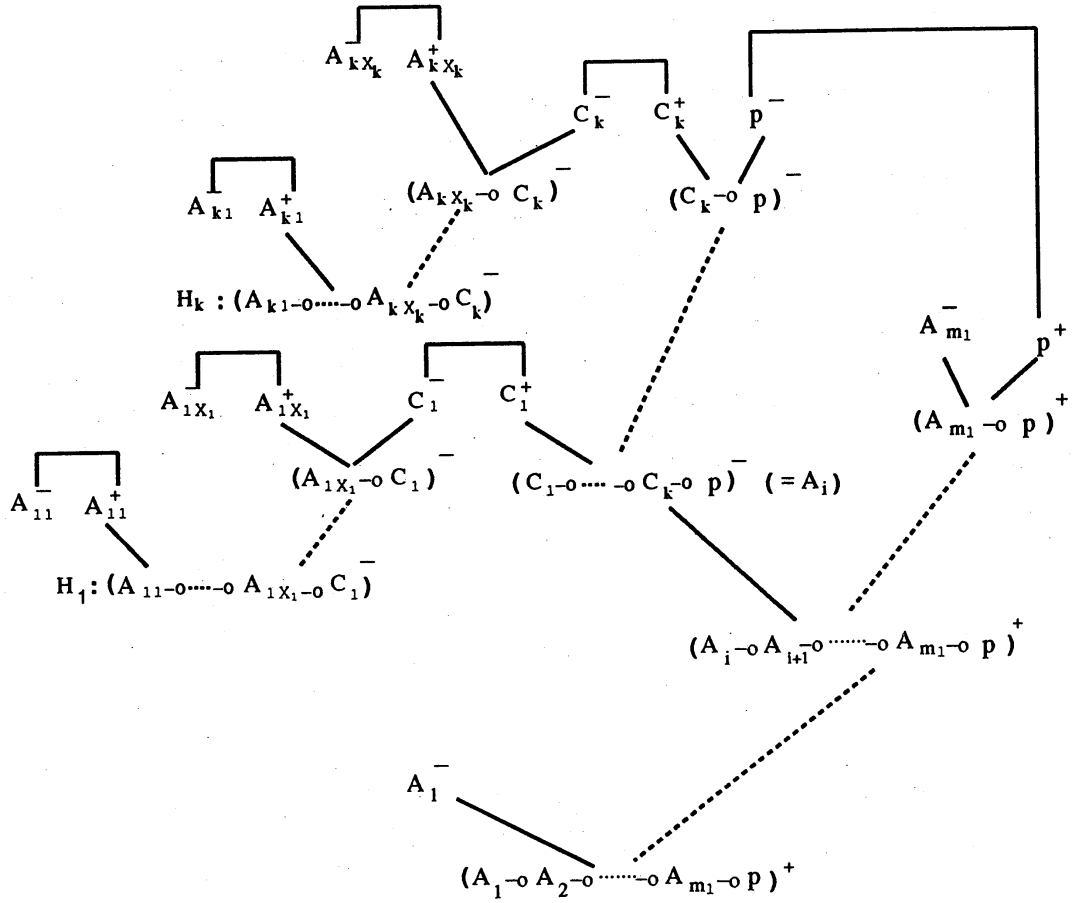


Fig. 8. a projection partial binding θ for Θ_1

Definition 14 (solved forms). Let $S = \{\langle \Theta_1^1, \Theta_2^1 \rangle, \ldots, \langle \Theta_1^n, \Theta_2^n \rangle\}$ be a system of IIMLL proof nets. A pair $\langle \Theta_1^j, \Theta_2^j \rangle$ for some $1 \leq j \leq n$ in S is in solved form if $\langle \Theta_1^j, \Theta_2^j \rangle = \langle \bar{\eta}(A, F), \Theta \rangle$ and $F \notin \mathcal{FPN}(S - \langle \bar{\eta}(A, F), \Theta \rangle)$, or both Θ_1 and Θ_2 have the unique one argument of $\top$-box. S is a solved system if each proof net pair $\langle \Theta_1^i, \Theta_2^i \rangle$ $(1 \leq i \leq n)$ in S is in solved form.

Definition 15 (higher order unification algorithm). Let S be a system of IIMLL proof nets. We have the following transformation. The following cases are chosen non-deterministically except for some restrictions that are stated in the below definition.

1. Case that S contains a pair of the same proof nets Θ **(removal):**

$$\{\langle \Theta, \Theta \rangle\} \cup S' \Longrightarrow S'.$$

2. Case that S contains a pair $\langle \Theta_1, \Theta_2 \rangle$ that is a solved pair (let us assume Θ_2 is the $\bar{\eta}$-normal form of an ID-link). Then Θ_2 has exactly one free port. Let the name of the free port be F) **(variable elimination):**

$$\{\langle \Theta_1, \Theta_2 \rangle\} \cup S' \Longrightarrow \{\langle \Theta_1, \Theta_2 \rangle\} \cup \{\Theta_1'\} S'.$$

Then, $\{\Theta_1'\}$ is the substitution by the transformation, where Θ_1' is Θ_1 except that the principal port of Θ_1' has the name F.
Note that $\{\Theta_1'\}$ only applies to one solved pair because of linearity of free ports.

3. Case that S contains a pair $\langle \Theta_1, \Theta_2 \rangle$ in which the heads of both proof nets have the same type and the following cases for the heads apply:
   (a) Case that both heads are the same constants, bound ports or free ports (**SIMPLE**):
   $$\{\langle \Theta_1, \Theta_2 \rangle\} \cup S' \Longrightarrow \mathrm{SIMPLE}(\langle \Theta_1, \Theta_2 \rangle) \cup S'.$$
   (b) Case that both heads are different constants:
   return 'failure'
   (c) Case that one head is a bound port and the other is a constant:
   return 'failure'
4. Case that there is a pair $\langle \Theta_1, \Theta_2 \rangle$ in which both heads of the proof nets have different type, or free ports with the same type.
   The following cases for the heads apply:
   (a) Case that one head is a constant and the other is a bound port:
   return 'failure'
   (b) Case that $\langle \Theta_1, \Theta_2 \rangle$ is not a solved pair:
      i. **(flexible-rigid pair)**
      Case that one head is a bound port or constant (let the proof net be Θ_2) and the other is a formula A with a free port name F (in Θ_1):
      A partial (flexible-rigid imitation or projection) binding θ is substituted for the free port:
      $$\{\langle \Theta_1, \Theta_2 \rangle\} \cup S' \Longrightarrow \{\langle \theta, \bar{\eta}(A, F) \rangle, \langle \Theta_1, \Theta_2 \rangle\} \cup S'.$$
      ii. Case that both heads are free ports (**flexible-flexible pair**):
      Let the head of Θ_1 (resp. Θ_2) has a formula A (resp B) with a free port name F (resp. F). And let the unique partial binding of $\top$-box for Θ_1 (resp. Θ_2) be θ_1 (resp. θ_2).
      $$\{\langle \Theta_1, \Theta_2 \rangle\} \cup S' \Longrightarrow \{\langle \theta_1, \bar{\eta}(A, F) \rangle, \langle \theta_2, \bar{\eta}(B, G) \rangle, \langle \Theta_1, \Theta_2 \rangle\} \cup S'.$$

In a sequence of transformations $S \Longrightarrow_* S'$ (where $\Longrightarrow_*$ is the reflexive-transitive closure of $\Longrightarrow$), if $S' = \{\langle \Theta_1^1, \Theta_2^1 \rangle, \ldots, \langle \Theta_1^n, \Theta_2^n \rangle\}$ is a solved system, then the procedure successfully terminates. Then let $\sigma_{S'}$ be the substitution by following way from S': each solved pair $\langle \Theta_1^i, \Theta_2^i \rangle$ in S', let Θ_2^i be a $\bar{\eta}$-normal form of an ID-link. Then Θ_2^i has exactly one free port. Let the name of the free port be F. Let Θ'^i_1 be Θ_1 except that the principal port of Θ_1' has the name F. $\sigma_{S'}$ is $\{\Theta_i' \mid 1 \leq i \leq n\}$.

Theorem 16 (Termination). *If all the proof nets in a system S are IIMLL proof nets, then the algorithm starting from S terminates.*

Proof. We can define the complexity measure $d(S)$ for a system S in the following way: let the multiset of the non-solved pairs in S be $\{\langle \Theta_1^1, \Theta_2^1\rangle, \ldots, \langle \Theta_1^n, \Theta_2^n\rangle\}$. $d(S) = \sum_{i=1}^{n}(d(\Theta_1^i) + d(\Theta_2^i))$.
Then, in any transformation of the above algorithm $S_1 \Longrightarrow S_2$, we can easily prove $d(S_2)$ is less than $d(S_1)$. We show this casewisely:

1. The case of **removal:**
   $d(S_2)$ is $2d(\Theta)$ smaller than $d(S_1)$.
2. The case of **variable elimination:**
   From our assumption of unification problems from which we start the procedure about free port names, it is obvious that in IIMLL, if a free port name occurs in a proof net system S_1 twice, then the free port name does not occur in the free port names in S_0 from which we start the procedure and is introduced by a partial binding. Hence at least one of two occurrences occurs in a solved pair in S_1. Thus the substitution with only one proof net in variable elimination applies to a solved pair and does not affect the increase of the complexity measure. By the way, $\langle \Theta_1, \Theta_2\rangle$ in variable elimination become in solved after the transformation. Hence $d(S_2)$ is $d(\Theta_1) + d(\Theta_2)$ smaller than $d(S_1)$.
3. The case of **SIMPLE:**
   Let both Θ_1 and Θ_2 has the number n of arguments. Then $d(S_2)$ is $2n$ smaller than $d(S_1)$.
4. The case of **flexible-flexible pair:**
   Since $\langle \Theta_1, \Theta_2\rangle$ become in solved, two new introduced proof nets pairs become in solved soon by variable elimination and these pairs affect only $\langle \Theta_1, \Theta_2\rangle$ in substitutions, $d(S_2)$ is $d(\Theta_1) + d(\Theta_2)$ smaller than $d(S_1)$.
5. The case of **flexible-rigid pair:**
   We must consider two cases.
   (a) The case of imitation binding:
   The new introduced pair $\langle \theta, \bar{\eta}(A, F)\rangle$ become in solve soon by variable elimination. The problem here is the different between $d(\langle \Theta_1, \Theta_2\rangle)$ and $d(\langle \theta\Theta_1, \Theta_2\rangle)$. Let $d(\Theta_1) = m_1 + d(e_1) + \cdots d(e_{m_1})$. Then since $d(\theta\Theta_1) = m_2 + m_1 + d(e_1) + \cdots d(e_{m_1})$, $d(\langle \theta\Theta_1, \Theta_2\rangle)$ may be greater than $d(\langle \Theta_1, \Theta_2\rangle)$. But note that after the application of the imitation binding, SIMPLE procedure must be applied to $d(\langle \theta\Theta_1, \Theta_2\rangle)$. Thus The problem here is reduced to the different between $d(\langle \Theta_1, \Theta_2\rangle)$ and $d(\mathrm{SIMPLE}(\langle \theta\Theta_1, \Theta_2\rangle))$. Then $d(\mathrm{SIMPLE}(\langle \theta\Theta_1, \Theta_2\rangle)) = (m_1 + d(e_1) + \cdots + d(e_{m_1})) + (d(\Theta_2) - m_2) < d(\Theta_1 + \Theta_2)$.
   (b) The case of projection:
   The new introduced pair $\langle \theta, \bar{\eta}(A, F)\rangle$ become in solve soon by variable elimination. The problem here is the different between $d(\langle \Theta_1, \Theta_2\rangle)$ and $d(\langle \theta\Theta_1, \Theta_2\rangle)$. Let $d(\Theta_1) = m_1 + d(e_1) + \cdots d(e_{m_1})$. And assume i-th argument e_i is projected. Since in the process of normalization of $\theta\Theta_1$

$e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{m_1}$ are appended to the bound ports of e_i with free port names H_1 and ... and H_k,
$d(\theta\Theta_1) = (m_1 - 1) + d(e_i) + d(e_1) + \ldots + d(e_{i-1}) + d(e_{i+1}) + \ldots + d(e_{m_1}) < d(\Theta_1)$

□

Theorem 17 (Soundness). *If for a given system S, the algorithm terminates in a system S', then S' is a solved pair and $\sigma_{S'}|_{\mathcal{FPN}(S)}$ is a unifier of S (where $\sigma_{S'}|_{\mathcal{FPN}(S)}$ is a substitution obtained from $\sigma_{S'}$ by restricting the domain of $\sigma_{S'}$ to $\mathcal{FPN}(S)$.*

Proof. Let the set of unifiers of a system S be $U(S)$. If the above algorithm transforms S into S', then it is easy to show that $U(S') \subseteq U(S)$. □

Definition 18 (pre-unifiers). Let $\cong$ be the least equivalence relation under the eight reductions and containing

$$\{\langle \Theta_1, \Theta_2 \rangle | \Theta_1 \text{ and } \Theta_2 \text{ has the same type and are both flexible proof nets } \}$$

defined in subsection 3.1.
A substitution θ is a pre-unifier of Λ_1 and Λ_2 with the same type if $\theta\Lambda_1 \cong \theta\Lambda_2$.

Theorem 19 (Completeness for pre-unification). *If θ is a pre-unifier of a system S, then there is a sequence of transformations $S \Longrightarrow_* S'$ with S' in solved form such that $\sigma_{S'}|\mathcal{FPN}(S)$ is a unifier of S.*

Proof. We can prove the same method as the approximation method in [SG89]. □

Remark. In the above higher order unification, the most general unifiers may not exist. Figure 9 shows a unification problem which has two solutions. One is a solution which substitute a imitation binding for H. The other is a solution which substitute a projection binding for H.

4 Related Works

In [CP97], higher order pre-unificatioan algorithm for an intuitionistic fragment of Linear Logic is studied independently of ours. However, the algororithm in [CP97] is based on a kind of inference rules, not on proof nets. Moreover, compared with standard Linear Logic, their system is very strange. The following unification problem is exemplified in [CP97]:

$$x : A, y : B \vdash F \hat{} x \hat{} y = c \hat{} (G_1 x y) \hat{} (G_2 x y) : a$$

where $\hat{}$ is the symbol of linear application. This problem has four solutions. This system is different from standard Linear Logic since a term is used both linearly and intuitionistically. But our unification problem based on standard

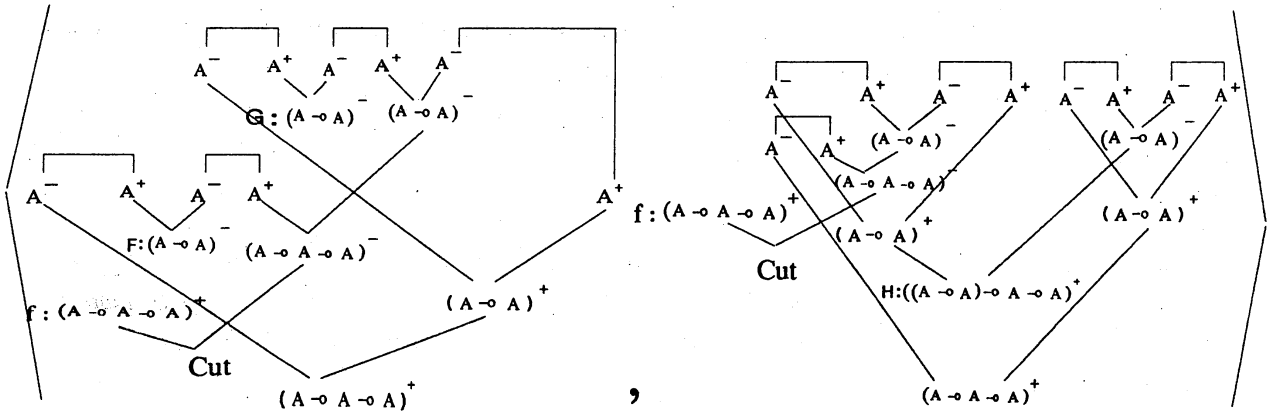


Fig. 9. an example of unification problem with several solutions

Linear Logic does not have such a example. For example the following problems exist in our system (if we borrow notation from them):

$$x : A, y : B \vdash F \hat{} x \hat{} y = c \hat{} (G_1 x) \hat{} (G_2 y) : a$$

$$x : A, y : B \vdash F \hat{} x \hat{} y = c \hat{} (G_1 x y) \hat{} G_2 : a$$

But these problems just has the unique solution, which has an advantage for practical applications since higher order unification usually is suffering from nondeterminism.

References

[ACCL91] M.Abadi, L.Cardelli, P.-L.Curien and J.-J.Levy. Explicit substitutions. *Journal of Functional Programming*, 1:375-416, 1991.

[CP97] I. Cervesato and F.Pfenning. Linear Higher-Order Pre-Unification. LICS'97, 1997.

[DHK95] G. Dowek, T. Hardin and C. Kirchner. Higher-order unification via explicit substitutions. LICS'95, 1995.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir95b] J.-Y. Girard. Proof-nets: the parallel syntax for proof-theory. In Ursini and Agliano, editors, *Logic and Algebra, New York, Marcel Dekker*, 1995.

[Gir95c] J.-Y. Girard. Light Linear Logic. Available by ftp anonymous on `lmd.univ-mrs.fr`, in `pub/girard`, 1995.

[Hue75] G. Huet. A Unification Algorithm for Typed λ-Calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Laf95] Y. Lafont. From Proof-Nets to Interaction Nets. *Advances in Linear Logic, London Mathematical Society Lecture Notes Series 222*, 1995.

[Mat96] S.Matsuoka. Towards a Higher Order Unification Based on Proof Nets. Proceedings of the JICSLP'96 workshop on Multi-Paradigm of Logic Programming, Report No.96-28, Technische Universitaet Berlin, pp.157-166, 1996.

[Mil91] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497-536, 1991.

[Pau86] L.C.Paulson. Natural deduction as higher-order resolution. *Journal of Logic Programming*, 3:237-258, 1986.

[SG89] W. Snyder and J.H. Gallier. Higher Order Unification Revisited: Complete Sets of Transformations. *Journal of Symbolic Computation*, 8:101-140, 1989.

[Wol93] D.A. Wolfram. The Clausal Theory of Types. Cambridge University Press, 1993.

This article was processed using the LaTeX macro package with LLNCS style